PSis - Programação de Sistemas - 2018/2019
Exame 1ª Época, 12 de Junho de 2018, 18h30, Duração: 2h

Software Architectures [5.0]

1. [1.0 points]
- What is an **Architectural Patterns**?

2. [1.0 points]
- Describe two advantages of using **Architectural Patterns** when implementing a complex system?

3. [1.0 points]
Most of the project implementations (distributed concentration game) did not implemented a pure **client-server Architectural pattern**.
- Describe the **client-server Architectural pattern**
- Explain how the project implementation differed from this pattern (If you group implemented a pure client-server in the project describe its use in the project).

4. [1.0 points]
- Describe the **Layered Architecture pattern.**
- Describe how this architectural pattern was applied in the project.
- Present and describe two advantages of its use.

5. [1.0 points] In the project, the communication between the client and the server was performed using low level socket messages, but RPC could also be used.
- Describe what is RPC
- Give an example of how the interface between the server and client would be defined.

Testing / software correctness [5.0]

6. [1.0 points]
- What is **integration testing**?
- What class of errors this tests try to find?
- How this type of tests gains from the use of a layered architecture in a system?

7. [2.0 points] The teaching staff provided a set of functions to help the development of the project. One of such functions was the **void init_board(int dim)** function that would create a square board and fill it with the pairs of strings.
- Present **3 unitary tests** that could be applied to that module
- for each test
    - Describe the error that would be found,
    - Present the pseudo-code of the test
    - Describe how to validate the result after performing the test

8. [1.0 points] Security testing guarantees that a system is capable of handling attacks. The project could suffer from different attacks from malicious clients (for instance a BOT with bad behavior).

- Describe two possible security tests to guarantee the project security .
- For each test
  - identify the possible attack
  - what needed to be coded in the project to pass such test.

9. [1.0 points]

- Describe why sending a 32 bits integers through a INET socket without any transformation (as presented in the example) affects the compatibility of a system:

```
int n;
…
write(sock_fd, &n, sizeof(n));
```

- Describe what changes in the previous code should be made to guarantee compatibility between components that communicate through INET sockets.

Processes / Threads / IPC [4.0]

10. [1.0 point]

- Explain how unix **signals** are implemented and how they may be used in the context of a complex system.

11. Most of the project implementations used stream sockets, where each socket was processed by a different thread, and had the board shared between all those threads so that each thread could verify independently the status of the cards (UP, DOWN, …).

11.a. [1.0 points]
Still maintaining each thread processing one socket (one socket <-> one thread):

- Describe a different organization of the server so that only one thread would access de board (the board was a local variable on that thread and no other thread could access it)
- Describe the different processing nodes and communication mechanism of this new approach.

If necessary draw a schema of the server organization (representing the various threads and communication channels.

11.b. [1.0 points]

- Present and describe one advantage and one drawback of this new server organizations?

12. [1.0 point]

- Compare **pipes** and **unix domain stream sockets** with respect to:
  - Addresses
  - Communication isolation.
- Explain in detail the similarities or differences presented.

Synchronization [6.0]

13. [2.0 point] Suppose that a thread code has a bug and such thread dies inside a critical region before unlocking the locked mutex.

- Describe what happens to the mutex in such case.
- Describe what happens to the system in such case.

14. [2.0 point] Condition variables are a type of synchronization objects defined in POSIX.

- Describe when they should be used.
- Describe how they are used (with a pseudo-code example)

15. [2.0 point] In the project it was necessary to store the players in a list. Every time a new player connected, a new structure would be inserted into the list, and every time a board update was made, it would be forwarded to every player.

The following code presents a pseudo-code implementation of the two functions **insertPlayer** and **sendUpdateAllPlayers**.

insertPlayer	**sendUpdateAllPlayers**
newPlayer = initializePlayer()	aux = playerList;
	while(aux != NULL){
newPlayer-> next = playerList;	sendUpdate(aux, x, y, UP)
playerList = newPlayer;	aux = aux -> next;
	}

Supposing that the players were never removed from the list:

- Identify the race condition
- Describe what are the effects on the program when the race condition happens.
- Change the code so that the race condition is solved.